PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-056971
(43)Date of publication of application : 25.02.2000

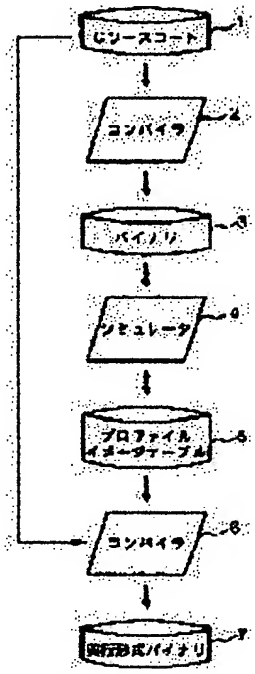
(51)Int.Cl. G06F 9/38
G06F 9/45

(21)Application number : 10-220329
(22)Date of filing : 04.08.1998

(71)Applicant : FUJITSU LTD
(72)Inventor : IWATA YASUSHI

(54) HIGH-SPEED ARITHMETIC PROCESSOR AND RECORDING MEDIUM

(57)Abstract:
PROBLEM TO BE SOLVED: To execute a program in data dependency relation fast by extracting information on data dependency relation and performing static prediction when a source program is compiled, and dynamically predicting and selectively executing an instruction which is not solved so far and has high prediction probability at the time of execution.
SOLUTION: A compiler 2 takes a morpheme analysis and a syntax analysis of a C source code 1 to generate a binary code. A simulator 4 registers in a profile image table 5 a pair of the line of an instruction in data dependency relation of an object to be predicted and a dependent instruction among instructions in data dependency relation. In the profile image table 5, the object instruction to be predicted among the instructions in the data dependency relation is registered. A compiler 6 generates a binary code 7 in execution form and registers an instruction code and a dependent instruction in the data dependence relation in the binary code in execution form corresponding to line numbers (address). The analysis and prediction of the source program and the fast execution of the compiled instruction in executable form are enabled.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－56971
（P2000－56971A）
(43)公開日　平成12年２月25日(2000.2.25)

(51)Int.Cl.⁷		識別記号	FI			テーマコード（参考）
Ｇ０６Ｆ	9/38	３５０	Ｇ０６Ｆ	9/38	３５０Ａ	５Ｂ０１３
	9/45			9/44	３２２Ｆ	５Ｂ０８１

審査請求　未請求　請求項の数８　ＯＬ　（全９頁）

(21)出願番号　特願平10－220329

(22)出願日　平成10年８月４日(1998.8.4)

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号

(72)発明者　岩田　靖
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

(74)代理人　100089141
弁理士　岡田　守弘

Ｆターム(参考)　5B013 AA00 AA12 CC00
5B081 AA06 CC11 CC21

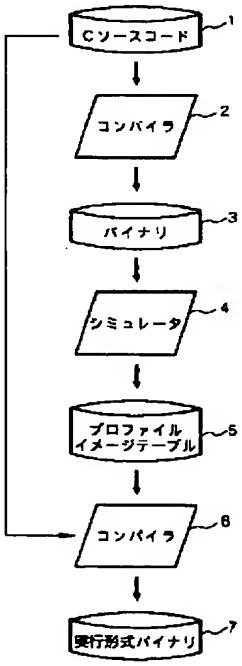
(54)【発明の名称】　高速演算処理装置および記録媒体

(57)【要約】

【課題】　本発明は、ソースプログラムを解析して予測を行う高速演算処理装置および記録媒体に関し、ソースプログラムのコンパイル時にデータ依存関係の情報を抽出して静的予測を行うと共に実行時にデータ依存関係があり実行時までに解決されなくて予測確率の高い命令を動的予測して選択実行し、データ依存関係にあるプログラムの高速実行を実現することを目的とする。

【解決手段】　ソースプログラムを解析する手段と、解析した結果をもとに行番号に対応づけてデータ依存関係にある命令をテーブルに設定する手段と、設定したデータ依存関係にある命令のうちデータ予測対象の命令の予測フラグをＯＮに設定する手段とを備えるように構成する。

本発明のシステム構成図



【特許請求の範囲】
【請求項１】ソースプログラムを解析して予測を行う高速演算処理装置において、
ソースプログラムを解析する手段と、
上記解析した結果をもとに行番号に対応づけてデータ依存関係にある命令をテーブルに設定する手段と、
上記設定したデータ依存関係にある命令のうちデータ予測対象の命令の予測フラグをＯＮに設定する手段とを備えたことを特徴とする高速演算処理装置。
【請求項２】上記予測フラグをＯＮに設定する命令として、自タスク内でデータが決定される命令を除いた命令としたことを特徴とする請求項１記載の高速演算処理装置。
【請求項３】上記予測フラグをＯＮに設定する命令として、他タスク内でデータが決定されるが命令実行時までに確定している命令を除いた命令としたことを特徴とする請求項１あるいは請求項２記載の高速演算処理装置。
【請求項４】ソースプログラムをコンパイルした実行可能形式の命令を高速実行する高速演算処理装置において、
実行可能形式の命令の実行時に、データ依存関係にある命令を設定したテーブルを参照してデータの値を予測して先行実行する手段と、
上記動的予測の結果の集計を行い予測の正解率の低い命令の先行実行を抑止する手段とを備えたことを特徴とする高速演算処理装置。
【請求項５】上記動的予測の結果の集計として、予測結果が正解のときに正解フラグに＋１、不正解のときに不正解フラグを＋１して正解率を集計することを特徴とする請求項４記載の高速演算処理装置。
【請求項６】上記データ依存関係にある命令として、メモリアクセス命令としたことを特徴とする請求項１から請求項５のいずれかに記載の高速演算処理装置。
【請求項７】ソースプログラムを解析する手段と、
上記解析した結果をもとに行番号に対応づけてデータ依存関係にある命令をテーブルに設定する手段と、
上記設定したデータ依存関係にある命令のうちデータ予測対象の命令の予測フラグをＯＮに設定する手段ととして機能させるプログラムを記録したコンピュータ読取可能な記録媒体。
【請求項８】実行可能形式の命令の実行時に、データ依存関係にある命令を設定したテーブルを参照してデータの値を予測して先行実行する手段と、
上記動的予測の結果の集計を行い予測の正解率の低い命令の先行実行を抑止する手段ととして機能させるプログラムを記録したコンピュータ読取可能な記録媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、ソースプログラムを解析して予測を行うおよびソースプログラムをコンパイルした実行可能形式の命令を高速実行する高速演算処理装置および記録媒体に関するものである。
【０００２】
【従来の技術】従来、プロセッサの実行性能向上を図るために分岐命令があるときに分岐先を予測して先行的に実行し、予測結果が正しければその先行実行した次から実行を継続し、予測結果が間違っていれば先行実行をキャンセルして他の分岐先を実行するようにしていた。
【０００３】
【発明が解決しようとする課題】上述したように従来は分岐先の予測、即ち制御の流れの分岐予測は行われていたが、あるタスク内で実行に必要なデータが他のタスク内で実行されて確定しないと実行できないといういわゆるデータ依存関係がある場合のデータ予測は行われていなく、高速実行し得ないという問題があった。
【０００４】本発明は、これらの問題を解決するため、ソースプログラムのコンパイル時にデータ依存関係の情報を抽出して静的予測を行うと共に実行時にデータ依存関係があり実行時までに解決されなくて予測確率の高い命令を動的予測して選択実行し、データ依存関係にあるプログラムの高速実行を実現することを目的としている。
【０００５】
【課題を解決するための手段】図１を参照して課題を解決するための手段を説明する。図１において、コンパイラ２は、Ｃソースコードを形態素解析、構文解析などを行い、バイナリを生成するものである。
【０００６】シミュレータ４は、データ依存関係にある命令のうち予測すべき命令を選択してプロファイルイメージテーブル５に登録するものである。プロファイルイメージテーブル５は、データ依存関係にある命令のうち予測すべき対象の命令を登録するものである（後述する図４参照）。
【０００７】実行形式バイナリ７は、実行形式のバイナリ（後述する図５参照）である。次に、動作を説明する。コンパイラ２がＣソースプログラムを解析し、シミュレータ４が解析した結果をもとに行番号に対応づけてデータ依存関係にある命令をプロファイルイメージジテーブル５に設定および設定したデータ依存関係にある命令のうちデータ予測対象の命令の予測フラグをＯＮに設定するようにしている。
【０００８】この際、予測フラグをＯＮに設定する命令として、自タスク内でデータが決定される命令を除いた命令とするようにしている。また、予測フラグをＯＮに設定する命令として、他タスク内でデータが決定されるが命令実行時までに確定している命令を除いた命令とするようにしている。
【０００９】また、実行可能形式の命令の実行時に、データ依存関係にある命令を設定したテーブルを参照してデータの値を予測して先行実行し、動的予測の結果の集

計を行い予測の正解率の低い命令の先行実行を抑止するようにしている。

【００１０】この際、動的予測の結果の集計として、予測結果が正解のときに正解フラグに＋１、不正解のときに不正解フラグを＋１して正解率を集計するようにしている。

【００１１】また、データ依存関係にある命令として、メモリアクセス命令とするようにしている。従って、ソースプログラムのコンパイル時にデータ依存関係の情報を抽出して静的予測を行うと共に実行時にデータ依存関係があり実行時までに解決されなくて予測確率の高い命令を動的予測して選択実行することにより、データ依存関係にあるプログラムを高速実行することが可能となる。

【００１２】

【発明の実施の形態】次に、図１から図１３を用いて本発明の実施の形態および動作を順次詳細に説明する。

【００１３】図１は、本発明のシステム構成図を示す。図１において、Ｃソースコードは、ソースコードの例であって、ここでは、Ｃ言語のソースコードである。

【００１４】コンパイラ２は、Ｃソースコード１を形態素解析、構文解析などを行い、バイナリ３を生成するものである。バイナリ３は、コンパイラ２によって生成された実行形式のコードである。

【００１５】シミュレータ４は、データ依存関係にある命令のうち予測すべき命令を選択してプロファイルイメージテーブル５に登録などするものである。プロファイルイメージテーブル５は、データ依存関係にある命令のうち予測すべき対象の命令を登録するものであって、後述する図４に示すようにデータ依存関係にある命令の行番号、依存命令の対を登録したものである。

【００１６】コンパイラ６は、実行形式バイナリ７を生成するものである。実行形式バイナリ７は、実行形式のバイナリであって、後述する図５に示すように行番号（アドレス）に対応づけて命令コード、データ依存関係にある依存命令を登録したものである。以下順次詳細に説明する。

【００１７】図２は、本発明の説明図（その１）を示す。図２において、ＰＥ（Ａ）、ＰＥ（Ｂ）は、プロセッサエレメントであって、タスクＡ、Ｂを動作させるものである。ここでは、ＰＥ（Ａ）上でタスクＡが動作し、ＰＥ（Ｂ）上でタスクＢが動作している。一点鎖線の矢印でデータ依存関係（１）、（２）、（３）を示し、このうちのデータ依存関係（１）のみが、データ予測して先行実行する対象である。

【００１８】ＰＣ（行番号）は、プログラムカウンタであって、現在の実行しようとする行番号（アドレス）を保持（生成）するものである。次に、図２の動作を説明する。

【００１９】（１）　ＰＥ（Ｂ）上で動作するタスクＢが行番号ＰＣ＝００１でｚ＝ｙ＋４の演算を行う。この際、演算中のｙはＰＥ（Ａ）で動作するタスクＡが行番号ＰＣ＝１０１でｙ＝ｃ＋２の演算によって算出されて確定するまで待機する必要があるというデータ依存関係がある。

【００２０】（２）　ＰＥ（Ｂ）上で動作するタスクＢが行番号ＰＣ＝００２でｚ＝ｚ＋２の演算を行う。この際、同じタスクＢで行番号ＰＣ＝００１で既に算出されて確定しており待機する必要がないというデータ依存関係がある。

【００２１】（３）　ＰＥ（Ｂ）で動作するタスクＢが行番号ＰＣ＝００３でｘ＝ｘ＋５の演算を行う。この際、演算中のｘはＰＥ（Ａ）で動作するタスクＡが行番号ＰＣ＝１００でｘ＝ａ＋１の演算によって算出されて確定しており待機する必要がないというデータ依存関係がある。

【００２２】従って、上記（１）、（２）、（３）のデータ依存関係のうち、待機する必要のあるデータ依存関係は（１）のみでありこれが本願発明のデータの値を予測して先行実行する対象の命令であり、後述する図５の実行形式バイナリの該当するアドレス（行番号）の予測フラグを１（予測）に設定する。予測する必要のないものには予測フラグを０（非予測）に設定する。これにより、コンパイル時にデータ依存関係にある命令のうち静的な予測が行われたこととなる。以上のことをまとめると以下のようになる。

【００２３】（イ）　自タスク内で定義されている命令は予測対象としない（図２の（２）の場合は予測対象としない）。

（ロ）　他タスク内で定義されていて、その命令を実行するまでにそのデータが確定している命令は予測対象としない（図２の（３）の場合は予測対象としない）。

【００２４】（ハ）　他タスク内で定義されていて、その命令を実行するときでも、そのデータが確定しない命令は予測対象とする（図２の（１）の場合は予測対象とする）。

【００２５】（ニ）　尚、後述する図８で説明するように、（ハ）の命令についてデータを予測して実際にシミュレーション実行して予測結果の正解率が高い命令のみを予測対象とし、正確率が低い命令を予測対象から外すようにしてもよい。

【００２６】図３は、本発明の説明図（その２）を示す。これは、図２のタスクＢ上で動作するソースイメージプログラムの例を示す。ここでは、例えば図２のタスクＢ上で実行される行番号００１は、００１　ａｄｄ　％ｒｚ，％ｒｙ，４となる。００１は行番号であり、ａｄｄは加算命令であり、％ｒｚ，％ｒｙ，４はレジスタｒｙの内容（ｙ）に４を加算してその結果をレジスタｒｚに格納するものである。同様に、行番号１２１、２００は、ＳＴ（ストア命令）、ｌｄ（ロード命令）につい

て記述したものである。

【００２７】図４は、本発明のプロファイルイメージテーブル例を示す。これは、データ依存関係にある命令を

プログラムカウンタ	依存命令１	依存命令２	依存命令３
００１	１０１	－	－

は、図２のタスクＢで実行される行番号００１の命令であって、データ依存関係の１番目が行番号１０１である旨を表す。データ依存関係の２番目、３番目はここではなしである。行番号２００の命令は、データ依存関係のある命令が１番目（依存命令１）１５０（レジスタｒ２）、２番目（依存命令２）１３０（レジスタｒ３）、３番目（依存命令３）１２１（レジスタｒ８）の３つがある。

アドレス	命令コード	依存命令１、依存命令２、依存命令３	予測フラグ
００１	add %rz,%ry,4　１０１	－　－	１（予測）
００２	add %rz,%rz,z　－	－　－	０（非予測）
００３	add %rx,%rx,5　１００	－　－	０（非予測）

以上のようにデータ依存関係にあるアドレス（行番号）、命令コード、依存命令１～３、予測フラグｆをコンパイル時に登録しておくことにより、実行時に予測フラグが１（予測）のみの命令についてデータの値を予測して先行実行することにより、選択的に静的予測した結果をもとに効率的に実行時にデータ依存関係にある命令を先行実行し、ＣＰＵの高速実行を実現することが可能となる。

【００３０】図６は、本発明の他のシステム構成図を示す。これは、実行時に後述する図８に示すように、ＰＣ（プログラムカウンタ）に設定されたアドレス（行番号）とタグとを比較して一致するときにヒット、一致するものみつからないときにミスヒットを出力し、ヒットのときは更に値と差分を読み出して演算器によって演算（ここでは、例えば加算）して予測値を算出するハードイメージである。

【００３１】以上の構成により、図５で予測フラグが１（予測）とした命令のアドレスとそのときの変数のデータの値、および前前回と前回との差分を高速に読み出し、データ依存関係にある変数の値を予測して先行実行することが可能となる。以下実行時の動作を順次詳細に説明する。

【００３２】図７は、本発明の説明図（その３、実行時）を示す。これは、ＰＣ（プログラムカウンタ）に行番号００１が設定され、図２のＰＥ（Ｂ）上で動作するタスクＢが行番号００１でｚ＝ｙ＋４を実行しようとしたときの様子を示す。

【００３３】初期値は、データ依存関係にある行番号００１の命令の変数ｙの初期値が０であることを表す。１回目は、データ依存関係にある行番号００１の命令の変数ｙの実行結果が８となったことを表す。

【００３４】２回目は、データ依存関係にある行番号００１の命令の変数ｙの予測の例を表す。（ａ）は、２回目に予測した値ｙ＝１６（１回目の値に差分を加算した８＋８＝１６）と実際に実行したｙの値１６とが一致して正解の場合を表す。この場合には、図示の下記のように更新する。

【００３５】

ＰＣ	値	差分
００１	１６	８

（ｂ）は、２回目に予測した値ｙ＝１６（１回目の値に差分を加算した８＋８＝１６）と実際に実行したｙの値８とが不一致で不正解の場合を表す。この場合には、図示の下記のように更新する。

【００３６】

ＰＣ	値	差分
００１	８	０

以上のように、初期値をもとに１回目の値を予測して実際の実行結果と一致したときにそのときの値および前回との差分を登録することを繰り返すと、後述する図８に示すようなテーブルを作成することが可能となる。そして、正解のときに予測した実行結果が正しいのでそれに続く処理を実行することが可能となり、一方、不正解のときに予測の結果をキャンセルして確定した変数の値で再実行する。これにより、実行時に正解となる値および差分が図８のテーブルに示すように登録され、当該正解数の割合が所定閾値（例えば６０％以上）の命令の予測を行い、それ以外の命令の予測を抑止することで、予測率の確率を向上させてキャンセル率を小さくして全体の実行速度を大幅に向上させることが可能となる。

【００３７】図８は、本発明の説明図（その４、実行時）を示す。これは、既述した図７の手順によって予測対象の命令（命令の行番号）について正解のときの変数の値および前回との差分を登録したリストであり、正解と不正解の数を集計して正解率が所定閾値（例えば６０％以上）の命令のみを残し他の命令の予測フラグを０

ソースプログラムから抽出して登録したものであって、図２の行番号００１、００２、００３のものを登録したものである。例えば

プログラムカウンタ	依存命令１	依存命令２	依存命令３
００１	１０１	－	－

【００２８】以上のように、コンパイル時に命令の行番号に対応づけてデータ依存関係のある他の命令の行番号を抽出して登録することにより、データ依存関係にある命令についてのみデータの値を予測して先行実行することが可能となる。

【００２９】図５は、本発明の実行形式バイナリ例を示す。これは、実行形式のバイナリであって、図示の下記の項目を登録したものである。

（非予測）に設定し、動的に統計を集計してデータ依存関係にあるデータの予測の正解率を向上させて高速実行させることを動的に制御することが可能となる。ここでは、ＰＣ＝００１（行番号００１）についてｎ回実行し、正解が１３９７４回、不正解が６４回となり、正解率が６０％以上であるので、データの依存関係にある命令（行番号００１の命令）のデータ予測を行う。一方、ＰＣ＝００２（行番号００２）についてｎ回実行し、正解が２１回、不正解が１７３６５回となり、正解率が６０％以下であるので、データの依存関係にある命令（行番号００２の命令）のデータ予測を抑止し、キャンセルによる不要な負荷による処理速度の低下を防止することを実行時に動的に行うことが可能となる。

【００３８】図９は、本発明の１実施例構成図を示す。図９の（ａ）は、実行オブジェクトの各ＰＥ＃１から＃４への割り振りの様子を示す。ここでは、図示のように、実行オブジェクトを４台のＰＥ＃１から＃４に割り振ると共に、このときに各アドレス（行番号）に対応づけて既述したデータ依存関係にある命令のアドレス（行番号）を依存命令１、２、３に登録しておく（コンパイル時に収集して登録しておく）。これら４台のＰＥ＃１から＃４にそれぞれ割り振られた実行オブジェクトは、逐次（ｂ）のＰＥ＃１から＃４によって実行する。

【００３９】図９の（ｂ）は、４台のＰＥ＃１から＃４によってそれぞれ実行オブジェクトを実行するときの処理を示す。ここでは、各ＰＥ＃１から＃４では、データ依存ありがＹＥＳ、依存解決済みがＮＯの場合には、変数の値を予測し、この予測した値をもとに実行オブジェクトをそれぞれ実行する。

【００４０】以上のように、実行オブジェクトを各ＰＥ＃１から＃４に割り振り、各ＰＥ＃１から＃４でデータ依存ありがＹＥＳ、依存解決済みがＮＯの場合に、変数の値を予測し、この予測した値をもとに実行オブジェクトをそれぞれ実行することにより、データ依存関係にある命令を正解率高く先行実行し、処理速度を高速化することが可能となる。

【００４１】図１０は、本発明の他の動作説明フローチャートを示す。これは、既述したデータ依存関係にある命令の変数の値の予測を動的に決定するときのフローチャートである。

【００４２】図１０において、Ｓ１は、依存ありか判別する。これは、例えば既述した図９の（ａ）で各ＰＥに割り振られた実行オブジェクトに依存命令１、２、３のいずれかが設定され、実行しようとする命令にデータ依存関係があるか判別する。ＹＥＳの場合には、データ依存関係があると判明したので、Ｓ２に進む。ＮＯの場合には、データ依存関係がないと判明したので、Ｓ５で予測フラグを０（非予測）にセットする。

【００４３】Ｓ２は、データ依存関係があったが解消したか判別する。ＹＥＳの場合には、データ依存関係が解消したのでＳ６で予測フラグを０（非予測）にセットする。ＮＯの場合には、データ依存関係が解消していないので、Ｓ３に進む。

【００４４】Ｓ３は、変数の値の予測した結果の正解率が高いか判別する。これは、命令の変数の値の予測を行ってその結果を既述した図８に示すように収集してその正解率が高い（例えば６０％以上）のときにデータ予測して高速化できると判明したので、Ｓ４で命令の変数のデータの値を予測する。一方、ＮＯの場合には、正解率が低く、キャンセルに伴う処理が発生し、予測を行うと処理速度がそのために低下してしまうので、Ｓ７で予測フラグを０（非予測）にセットする。

【００４５】以上のように、実行時に動的にデータの依存関係があり、データの依存関係が解消していなく、かつデータ予測したときの正解率が高いときに命令の変数のデータの値の予測を行い、それ以外の命令についてデータの予測を抑止し、命令の実行速度を動的に解析して効率的に高速化を図ることが可能となる。

【００４６】図１１は、本発明の予測正解率例を示す。これは、ベンチマークプログラムＳＣ（ＳＰＥＣｉｎｔ９２、System Perfomance Evaluation Cooperative)をシミュレート実行したときのデータであって、横軸が各種処理を行うプログラムなどをを表し、縦軸が予測正解率を表す。図中の棒グラフの横線の部分がＬＯＡＤ命令の部分を表し、横線の部分がＳＴＯＲＥ命令の部分を表し、点々の部分がＯＴＨＥＲＳ（ＬＡＡＤ、ＳＴＯＲＥなどのメモリアクセス命令以外の命令）の部分を表す。図中の記号は下記を表す。

【００４７】

ｃｏｍｐ：ｃｏｍｐｒｅｓｓ（圧縮プログラム）
ｅｑｎ　：ｅｑｎｔｏｔｔ（Ｃプログラム）
ｅｓｐ　：ｅｓｐｒｅｓｓｅ
ｇｃｃ　：コンパイラ
ｌｉ　　：ｌｉｓｐインタプリタで９クイーン問題を解く
ｓｃ　　：スプレッドシート
ａｖｅ　：平均

図１２は、本発明の選択的値予測の対象命令と非対象命令例（％）を表す。これは、図１１で既述したベンチマークプログラムＳＣを分類したものであって、図示の下記の項目に分類したものである。

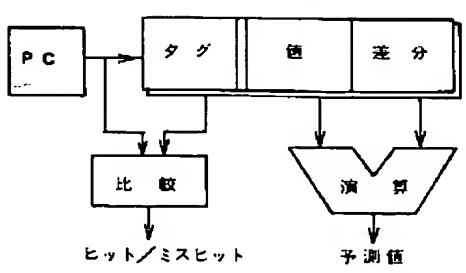
【００４８】

・依存無し（データ依存関係がない場合）：
・解決済み（データ依存関係はあるが実行時までに解決済みの場合）：
・予測対象（データ依存関係があり、解決済みでない場合）：本願発明の命令のデータの値の予測対象となる命令
・ｌｏａｄ：データをメモリにロードする命令
・ｓｔｏｒｅ：メモリからデータをレジスタにストアす

る命令
・ｏｔｈｅｒｓ：ｌｏａｄ、ｓｔｏｒｅのメモリアクセス命令以外の命令
ここで、左側に記載した（１）は、図２の（１）に対応して、データ依存関係を予測する対象の割合（％）である。
【００４９】（２）は、図２の（２）に対応して、データ依存関係がない命令の割合（％）である。（３）は、図２の（３）に対応して、データ依存関係があるか命令の実際の実行時までに解決してデータの値が確定している命令の割合（％）である。
【００５０】従って、ベンチマークプログラムＳＣでは、予測対象は、ｃｏｍｐ、ｅｑｎ、ｅｓｐ、ｇｃｃ、ｌｉ、ｓｃ、Ａｖｅで９．２～１１．６％の間の割合であり、これらについて選択的に命令の変数の値の予測を行い、むやみに予測する際のキャンセルに伴う処理の低下を無くし、予測したときに高速化される可能性の高いもののみのデータ依存関係にある命令の変数の値を予測して先行実行することが可能となる。更に、ｌｏａｄ命令やｓｔｏｒｅ命令のメモリアクセスは実行速度が遅く先行実行するメリットが高いので、当該メモリアクセス命令（ｌｏａｄ命令、ｓｔｏｒｅ命令）を予測対象として選択し、データ依存関係にある命令の変数の値を予測して先行実行するようにしてもよい。
【００５１】図１３は、本発明の実行時間例を示す。これは、既述したベンチマークプログラムＳＣの各プログラムをシミュレート実行したときの時間を表し、横軸が各種処理を行うプログラムなどを表し、縦軸が実行時間（Ｋサイクル）を表す。図中の棒グラフの左側が予測なし、中央が予測あり、右側が完全予測（予測が全部正解の場合）を表す。例えばｓｃの場合には、予測なしの場合に比して、本願発明の予測を行うと、２０．９％の時間（サイクル数）だけ高速化できることが判明した。詳細に説明すれば、ベンチマークプログラムＳＣ中に含まれる９．２％のｓｃ（スプレッドシート）の命令を予測対象とし、約２０．９％のサイクル数だけ高速化できることが判明した。この際、予測を全て正解とすると、約３４．５％のサイクル数だけ高速化できるとシミュレートできた。ベンチマークプログラム中の他のプログラムについても図１２に示す全体に対するそれぞれの割合に対し、図１３の中央の予測ありのサイクル数に高速化できる。
【００５２】
【発明の効果】以上説明したように、本発明によれば、ソースプログラムのコンパイル時にデータ依存関係の情報を抽出して静的予測を行うと共に実行時にデータ依存関係があり実行時までに解決されなくて予測確率の高い命令を動的予測して選択実行する構成を採用しているため、データ依存関係にあるプログラムを高速実行する装置を実現できる。
【図面の簡単な説明】
【図１】本発明のシステム構成図である。
【図２】本発明の説明図（その１）である。
【図３】本発明の説明図（その２）である。
【図４】本発明のプロファイルイメージテーブル例である。
【図５】本発明の実行形式バイナリ例である。
【図６】本発明の他のシステム構成図である。
【図７】本発明の説明図（その３、実行時）である。
【図８】本発明の説明図（その４、実行時）である。
【図９】本発明の１実施例構成図である。
【図１０】本発明の他の動作説明フローチャートである。
【図１１】本発明の予測正解率例である。
【図１２】本発明の選択的値予測の対象命令と非対象命令例である。
【図１３】本発明の実行時間例である。
【符号の説明】
１：ソースコード
２、６：コンパイラ
３：バイナリ
４：シミュレータ
５：プロファイルイメージテーブル
７：実行形式バイナリ

【図６】

本発明の他のシステム構成図



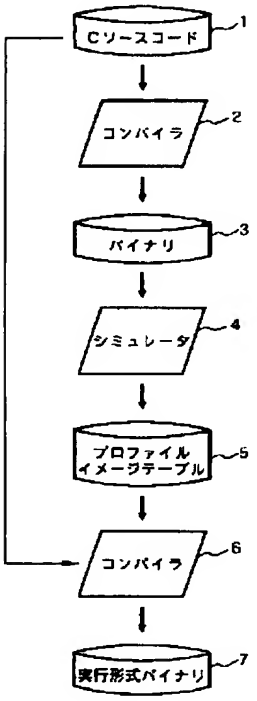
【図１２】

本発明の選択的値予測の対象命令と非対象命令例［％］

		comp	eqn	esp	gcc	li	sc	Ave.
(2)	依存無し	52.4	47.3	52.0	57.8	53.0	48.2	51.8
(3)	解決済	38.1	42.0	38.9	32.0	35.4	42.6	38.1
(1)	予測対象	9.5	10.7	9.1	10.2	11.6	9.2	10.2
	load	15.8	14.5	24.5	22.0	42.6	18.9	23.0
	store	31.8	14.7	19.7	14.7	13.7	13.5	18.0
	others	52.3	70.8	55.8	63.4	43.6	67.6	58.9

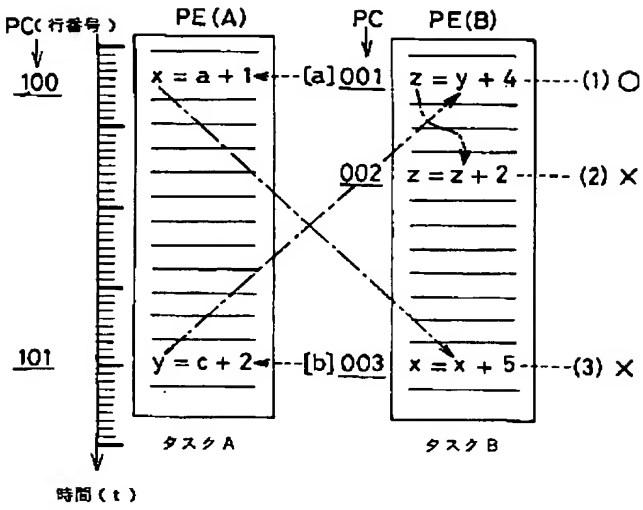
【図１】

本発明のシステム構成図



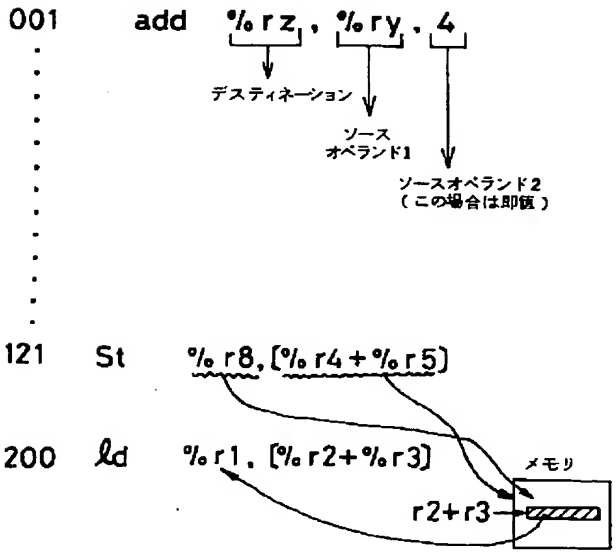
【図２】

本発明の説明図（その１）



【図３】

本発明の説明図（その２）



【図４】

本発明のプロファイルイメージテーブル例

プログラム カウンタ	依存 命令１	依存 命令２	依存 命令３
001	101	—	—
002	—	—	—
003	100	—	—
200	150(r2)	130(r3)	121(r8)

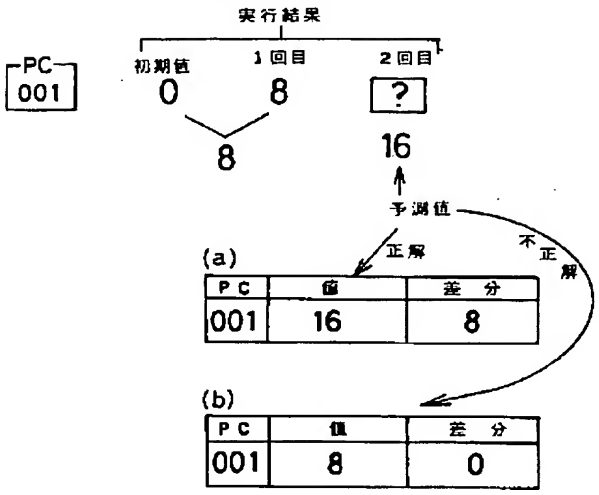
【図５】

本発明の実行形式バイナリ例

アドレス	命令コード	依存 命令１	依存 命令２	依存 命令３	予測フラグ
001	add %rz, %ry, 4	101	—	—	1
002	add %rz, %rz, 2	—	—	—	0
003	add %rx, %rx, 5	100	—	—	0
200	ld %r1, [%r2+%r3]	150(r2)	130(r3)	121(r8)	

【図７】

本発明の説明図（その３，実行時）



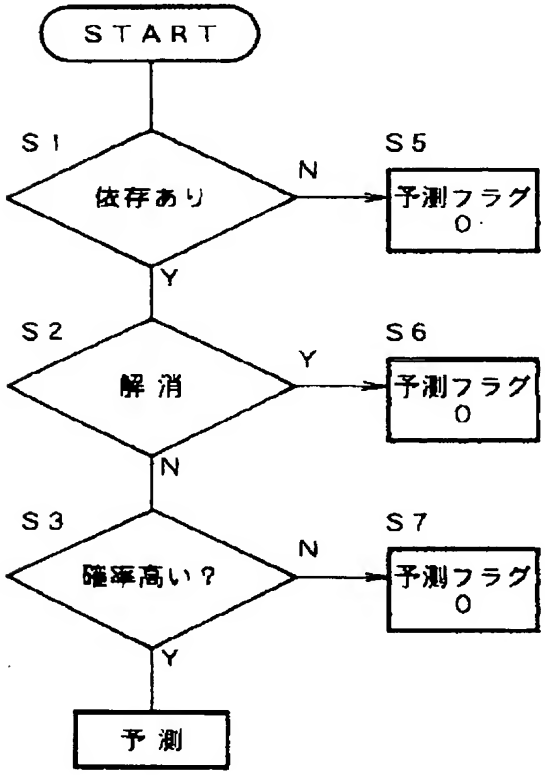
【図８】

本発明の説明図（その４，実行時）

	PC	値	差	正解	不正解
1回目	001	0	0	0	0
2回目 不正解	001	8	8	0	1
3回目 正解	001	16	8	1	1
4回目 不正解	001	16	0	1	2
	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・
n回	001	32	2	13974	64
	⋮				
n回	002	64	4	21	17365

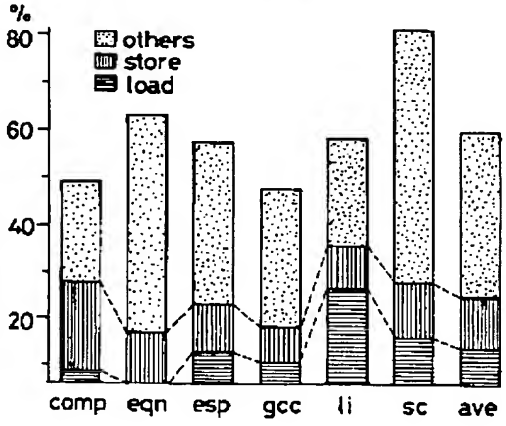
【図１０】

本発明の他の動作説明フローチャート



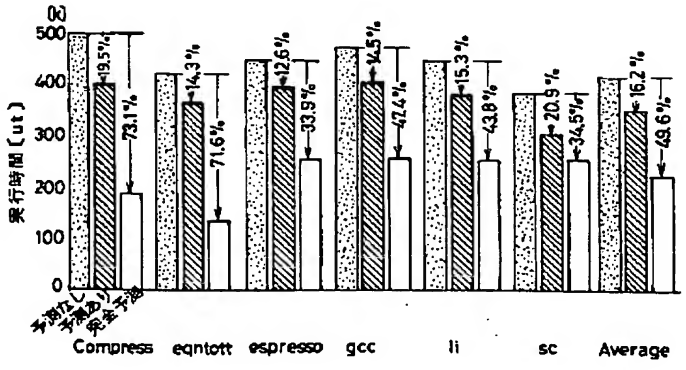
【図１１】

本発明の予測正解率例



【図１３】

本発明の実行時間例



【図９】

本発明の１実施例構成図